Package: n4m (via r-universe)

June 12, 2026

Type Package

Title Portable Partial Least Squares and NIRS Engine

Version 0.99.0

Date 2026-05-20

Description Implements a portable Partial Least Squares (PLS) and Near-Infrared Spectroscopy (NIRS) engine. Provides fit/predict wrappers for the shipped PLS regression solvers (NIPALS, SIMPLS, SVD, kernel, wide-kernel, orthogonal-scores, power, randomized SVD, PCR), variants (sparse SIMPLS, CPPLS, weighted, robust, ridge, continuum, multi-block, GLM, MIR), adaptive AOM-PLS and POP-PLS operator selection, variable-selection methods (SPA, CARS, GA, random frog, stability selection, VIP), diagnostics (Hotelling T2, Q residuals, DModX), and calibration transfer (PDS, DS). The C++17 implementation is vendored and compiled from source at install time; no external system libraries are required.

License CeCILL (== 2.1)

URL https://github.com/GBeurier/nirs4all-methods

BugReports https://github.com/GBeurier/nirs4all-methods/issues

SystemRequirements C++17, GNU make

NeedsCompilation yes

Encoding UTF-8

Imports stats

Suggests testthat (>= 3.0.0), knitr, rmarkdown, markdown, pls

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs make

Repository https://gbeurier.r-universe.dev

Date/Publication 2026-06-12 05:27:52 UTC

RemoteUrl https://github.com/GBeurier/nirs4all-methods

RemoteRef HEAD

RemoteSha 288e2a888054663484f332ee2a69209c2bb071d2

RemoteSubdir bindings/r/n4m

Contents

`aom_pls` *AOM-PLS and POP-PLS adaptive operator selection*

Description

`aom_pls()` (alias `aompls()`) runs global Adaptive Operator Mixture PLS selection: one preprocessing operator is selected for the whole PLS model. `pop_pls()` (alias `poppls()`) runs per-component operator selection, where each retained PLS component may use a different preprocessing operator.

Usage

```
aom_pls(X, Y, max_components = 3L, n_operators = 9L, cv = 3L)

aompls(X, Y, max_components = 3L, n_operators = 9L, cv = 3L)

pop_pls(X, Y, max_components = 3L, n_operators = 9L, cv = 3L)

poppls(X, Y, max_components = 3L, n_operators = 9L, cv = 3L)
```

Arguments

`X`	Numeric matrix of spectra, with rows as samples and columns as wavelengths.
`Y`	Numeric vector or matrix of responses, with one row per sample.
`max_components`	Maximum number of latent PLS components.
`n_operators`	Number of compact AOM bank operators to expose.
`cv`	Number of contiguous cross-validation folds.

Details

The operator bank mirrors the compact public `nirs4all` AOM bank: identity, Savitzky-Golay smoothers, Savitzky-Golay derivatives, polynomial detrending, and finite difference. The implementation is provided by the native n4m AOM selector ABI.

Value

A named list. Both functions return `predictions`, `operator_kinds`, component-selection diagnostics, cross-validation scores, and selected component metadata.

Examples

```
set.seed(1)
X <- matrix(rnorm(40 * 20), nrow = 40)
Y <- as.numeric(X[, 1] + rnorm(40, sd = 0.1))
fit <- aom_pls(X, Y, max_components = 2L)
dim(fit$predictions)
fit2 <- pop_pls(X, Y, max_components = 2L)
dim(fit2$predictions)
```

`aom_preprocess` *Adaptive Operator-Mixture preprocessing fit/transform.*

Description

Fits the AOM preprocessing pipeline (operator bank + gating) over ‘X‘ and returns a ‘n4m_method_fit‘ object carrying the selected operators, the per-component gating weights, and the transformed spectra ready to feed into a downstream regression solver.

Usage

```
aom_preprocess(X, Y = NULL, n_operators = 1L, gating_mode = 0L)
```

Arguments

`X`	numeric matrix of spectra (rows = samples, cols = wavelengths).
`Y`	optional numeric vector of supervisory targets. When ‘NULL‘, the unsupervised gating path is used (a zero target vector is substituted internally).
`n_operators`	number of operators in the AOM bank (default ‘1L‘).
`gating_mode`	integer code selecting the gating strategy: ‘0L‘ = hard-select per component, ‘1L‘ = soft-mixture (default ‘0L‘).

Value

A ‘n4m_method_fit‘ object. Use ‘predict()‘ for inference on new spectra and ‘coef()‘ to extract the gating coefficients.

Examples

```
set.seed(1)
X <- matrix(rnorm(40 * 10), nrow = 40)
Y <- as.numeric(X[, 1] + rnorm(40, sd = 0.1))
fit <- aom_preprocess(X, Y)
class(fit)
```

approximate_press *Approximate-PRESS component selection.*

Description

For each component count k in 1..max_components, fits SIMPLS and approximates PRESS via leverage-inflated in-sample residuals.

Usage

```
approximate_press(X, Y, max_components)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
max_components	Integer; maximum number of components to scan.

Value

A list with ‘press_per_component‘, ‘rmse_per_component‘, ‘selected_n_components‘ (the argmin of PRESS, 1-based as an integer length 1).

bagging_pls *Bagging PLS — formula entry point.*

Description

Bagging PLS — formula entry point.

Usage

```
bagging_pls(
  formula,
  data,
  ncomp = 2L,
  n_estimators = 50L,
  seed = 0L,
  na.action = stats::na.omit
)
```

Arguments

formula	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
data	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
ncomp	Integer. Number of latent components.
n_estimators	Integer >= 1. Number of bootstrap / boosting / random-subspace estimators.
seed	Integer. Random seed for reproducibility.
na.action	What to do with ‘NA‘s. Default: ‘na.omit‘.

Value

A ‘n4m_method_fit‘ object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use ‘predict()‘ for inference and ‘coef()‘ to extract regression coefficients.

---

bagging_pls_fit *Bagging PLS (bootstrap aggregation of PLS regressors).*

Description

Bagging PLS (bootstrap aggregation of PLS regressors).

Usage

```
bagging_pls_fit(X, Y, n_components, n_estimators = 50L, seed = 0L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_estimators	Integer >= 1. Number of bootstrap / boosting / random-subspace estimators.
seed	Integer. Random seed for reproducibility.

bipls_select *biPLS — backward interval PLS.*

Description

biPLS — backward interval PLS.

Usage

```
bipls_select(X, Y, n_components, interval_width = 10L, min_intervals = 1L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
interval_width	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
min_intervals	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

boosting_pls *Boosting PLS — formula entry point.*

Description

Boosting PLS — formula entry point.

Usage

```
boosting_pls(
  formula,
  data,
  ncomp = 2L,
  n_estimators = 50L,
  learning_rate = 0.1,
  na.action = stats::na.omit
)
```

Arguments

formula	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
data	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
ncomp	Integer. Number of latent components.
n_estimators	Integer >= 1. Number of bootstrap / boosting / random-subspace estimators.
learning_rate	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
na.action	What to do with ‘NA‘s. Default: ‘na.omit‘.

Value

A ‘n4m_method_fit‘ object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use ‘predict()‘ for inference and ‘coef()‘ to extract regression coefficients.

---

boosting_pls_fit *Boosting PLS (stage-wise refit with learning_rate).*

Description

Boosting PLS (stage-wise refit with learning_rate).

Usage

```
boosting_pls_fit(X, Y, n_components, n_estimators = 50L, learning_rate = 0.1)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_estimators	Integer >= 1. Number of bootstrap / boosting / random-subspace estimators.
learning_rate	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.

bve_select *BVE-PLS.*

Description

BVE-PLS.

Usage

```
bve_select(X, Y, n_components, n_steps = 10L, min_features = 5L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_steps	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
min_features	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

cars_select *CARS — Competitive Adaptive Reweighted Sampling.*

Description

Fit-data selector with a built-in 5-fold validation plan (default C-side fallback). For a custom plan, use the lower-level Python binding or extend r_methods.c.

Usage

```
cars_select(X, Y, n_components, n_iterations = 50L, min_features = 5L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_iterations	Number of CARS iterations (typical 50-100).
min_features	Lower bound on the final subset size.

Value

A list with 'selected_indices' (1-based) and 'best_rmse'.

coef.n4m_fit *Extract the regression coefficients of a [pls()]-fitted model.*

Description

Returns the ‘(p x q)‘ coefficient matrix (‘p‘ predictors by ‘q‘ targets) read from the fitted libn4m model via the ‘n4m_model_get_array‘ C ABI (tag ‘N4M_MODEL_COEFFICIENTS‘). Rows are named after the predictors when their names are available from the model terms.

Usage

```
## S3 method for class 'n4m_fit'
coef(object, ...)
```

Arguments

object	A ‘n4m_fit‘ returned by [pls()].
...	Ignored (for S3 generic compatibility).

Value

A numeric ‘p x q‘ matrix of regression coefficients.

coefficient_select *Coefficient-magnitude ranker.*

Description

Coefficient-magnitude ranker.

Usage

```
coefficient_select(model, X, top_k)
```

Arguments

model	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
X	Numeric matrix used for the fit (re-passed for diagnostics).
top_k	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.

Value

A list with ‘scores‘ (|coef| sums) and ‘selected_indices‘.

`continuum_regression` *Continuum regression — formula entry point.*

Description

Continuum regression — formula entry point.

Usage

```
continuum_regression(
  formula,
  data,
  ncomp = 2L,
  tau = 0.5,
  na.action = stats::na.omit
)
```

Arguments

`formula`	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
`data`	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
`ncomp`	Integer. Number of latent components.
`tau`	Numeric in [0, 1]. Continuum regression mixing parameter.
`na.action`	What to do with ‘NA‘s. Default: ‘na.omit‘.

Value

A ‘n4m_method_fit‘ object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use ‘predict()‘ for inference and ‘coef()‘ to extract regression coefficients.

`continuum_regression_fit` *Continuum regression (tau in [0, 1]).*

Description

Continuum regression (tau in [0, 1]).

Usage

```
continuum_regression_fit(X, Y, n_components, tau = 0.5)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_components`	Integer. Number of latent components.
`tau`	Numeric in [0, 1]. Continuum regression mixing parameter.

`cppls` *Canonical Powered PLS — formula entry point.*

Description

Canonical Powered PLS — formula entry point.

Usage

```
cppls(formula, data, ncomp = 2L, gamma = 0.5, na.action = stats::na.omit)
```

Arguments

`formula`	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
`data`	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
`ncomp`	Integer. Number of latent components.
`gamma`	Numeric. CPPLS / kernel band parameter.
`na.action`	What to do with ‘NA‘s. Default: ‘na.omit‘.

Value

A ‘n4m_method_fit‘ object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use ‘predict()‘ for inference and ‘coef()‘ to extract regression coefficients.

cppls_fit *Canonical Powered PLS fit (Indahl 2005).*

Description

Canonical Powered PLS fit (Indahl 2005).

Usage

```
cppls_fit(X, Y, n_components, gamma = 0.5)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
gamma	Power exponent in [0, 1]. 0 recovers SIMPLS, 1 is fully power-rescaled.

Value

A list with ‘coefficients‘, ‘predictions‘, ‘x_mean‘, ‘y_mean‘, ‘rmse‘.

di_pls *Domain-invariant PLS – formula entry point.*

Description

Domain-invariant PLS – formula entry point.

Usage

```
di_pls(
  formula,
  data,
  ncomp = 2L,
  X_target,
  di_lambda = 1,
  na.action = stats::na.omit
)
```

Arguments

formula	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
data	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
ncomp	Integer. Number of latent components.
X_target	Numeric matrix for the target domain.
di_lambda	Numeric DI-PLS penalty.
na.action	What to do with ‘NA‘s. Default: ‘na.omit‘.

Value

A ‘n4m_method_fit‘ object carrying the fitted model and in-sample predictions.

---

di_pls_fit *Domain-Invariant PLS (Nikzad-Langerodi 2018).*

Description

Domain-Invariant PLS (Nikzad-Langerodi 2018).

Usage

```
di_pls_fit(X_source, Y_source, n_components, X_target, di_lambda = 1)
```

Arguments

X_source	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
Y_source	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
n_components	Integer. Number of latent components.
X_target	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
di_lambda	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.

ds_fit *Direct Standardization (calibration transfer).*

Description

Direct Standardization (calibration transfer).

Usage

```
ds_fit(X_source, X_target)
```

Arguments

X_source	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
X_target	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

ecr *Elastic Component Regression — formula entry point.*

Description

Elastic Component Regression — formula entry point.

Usage

```
ecr(formula, data, ncomp = 2L, alpha = 0.5, na.action = stats::na.omit)
```

Arguments

formula	A two-sided formula (e.g. 'y ~ .' or 'y ~ x1 + x2 + x3'). Response on the left, predictors on the right.
data	A 'data.frame' (or anything 'as.data.frame'-coercible) containing the response and predictor columns referenced by 'formula'.
ncomp	Integer. Number of latent components.
alpha	Numeric in [0, 1]. Elastic-net / penalty mixing parameter.
na.action	What to do with 'NA's. Default: 'na.omit'.

Value

A 'n4m_method_fit' object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use 'predict()' for inference and 'coef()' to extract regression coefficients.

ecr_fit *Elastic Component Regression (Liu 2009/2010).*

Description

Elastic Component Regression (Liu 2009/2010).

Usage

```
ecr_fit(X, Y, n_components, alpha = 0.5)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
alpha	Numeric in [0, 1]. Elastic-net / penalty mixing parameter.

emcuve_select *EMCUVE — ensemble Monte Carlo UVE.*

Description

EMCUVE — ensemble Monte Carlo UVE.

Usage

```
emcuve_select(
  X,
  Y,
  n_components,
  noise_features = NULL,
  noise_seed = 0L,
  n_ensembles = 5L,
  vote_threshold = 0.5
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
noise_features	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

noise_seed	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
n_ensembles	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
vote_threshold	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

`fused_sparse_pls_fit` *Fused-sparse PLS (L1 + adjacent-coef smoothing).*

Description

Fused-sparse PLS (L1 + adjacent-coef smoothing).

Usage

```
fused_sparse_pls_fit(
  X,
  Y,
  n_components,
  l1_lambda = 0.05,
  fusion_lambda = 0.05
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
l1_lambda	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
fusion_lambda	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

`ga_select` *GA-PLS — genetic algorithm variable selection.*

Description

GA-PLS — genetic algorithm variable selection.

Usage

```
ga_select(
  X,
  Y,
  n_components,
  n_generations = 50L,
  population_size = 50L,
  min_features = NULL,
  max_features = NULL,
  mutation_rate = 0.01,
  seed = 0L
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_generations	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
population_size	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
min_features	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
max_features	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
mutation_rate	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
seed	Integer. Random seed for reproducibility.

---

gpr_pls_fit *Gaussian Process Regression on PLS scores (single-target Y).*

Description

Gaussian Process Regression on PLS scores (single-target Y).

Usage

```
gpr_pls_fit(
  X,
  Y,
  n_components,
```

```
  length_scale = 1,
  noise_level = 1e-04,
  seed = 0L
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
length_scale	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
noise_level	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
seed	Integer. Random seed for reproducibility.

---

group_sparse_pls_fit *Group-sparse PLS (group L1 across feature groups).*

Description

Group-sparse PLS (group L1 across feature groups).

Usage

```
group_sparse_pls_fit(X, Y, n_components, group_assignment, group_lambda = 0.05)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
group_assignment	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
group_lambda	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

interval_select *Interval selector (iPLS).*

Description

Interval selector (iPLS).

Usage

```
interval_select(X, Y, n_components, interval_width = 10L, step = 1L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
interval_width	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
step	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

ipw_select *IPW-PLS.*

Description

IPW-PLS.

Usage

```
ipw_select(
  X,
  Y,
  n_components,
  n_iterations = 10L,
  top_k = 10L,
  damping = 0.5,
  weight_floor = 1e-06
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_iterations	Integer >= 1. Number of outer-loop iterations.
top_k	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
damping	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
weight_floor	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

---

irf_select *IRF — Interval Random Frog.*

Description

IRF — Interval Random Frog.

Usage

```
irf_select(
  X,
  Y,
  n_components,
  n_iterations = 100L,
  window_size = 10L,
  initial_intervals = 10L,
  top_k = 5L,
  seed = 0L
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_iterations	Integer >= 1. Number of outer-loop iterations.
window_size	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
initial_intervals	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

top_k	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
seed	Integer. Random seed for reproducibility.

`iriv_select` *IRIV — Iteratively Retains Informative Variables.*

Description

IRIV — Iteratively Retains Informative Variables.

Usage

```
iriv_select(X, Y, n_components, max_rounds = 20L, seed = 0L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
max_rounds	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
seed	Integer. Random seed for reproducibility.

`kennard_stone_split` *Kennard-Stone train/test split*

Description

Thin R wrapper over libn4m's Kennard-Stone splitter C ABI. The R layer performs matrix layout conversion only; all split logic is delegated to libn4m.

Usage

```
kennard_stone_split(X, test_size = 0.25, zero_based = FALSE)
```

Arguments

X	Numeric matrix.
test_size	Fraction of samples assigned to the test set.
zero_based	Logical; return 0-based indices when TRUE. The default FALSE returns idiomatic 1-based R indices.

Value

Named list with integer vectors `train` and `test`.

kernel_pls_fit *Non-linear kernel PLS (Rosipal & Trejo 2001).*

Description

Non-linear kernel PLS (Rosipal & Trejo 2001).

Usage

```
kernel_pls_fit(
  X,
  Y,
  n_components,
  kernel_type = 1L,
  gamma = 0,
  coef0 = 1,
  degree = 3L
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
kernel_type	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
gamma	Numeric. CPPLS / kernel band parameter.
coef0	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
degree	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

lw_pls_fit *Locally-weighted PLS (Næs & Centner 1998).*

Description

Locally-weighted PLS (Næs & Centner 1998).

Usage

```
lw_pls_fit(X, Y, n_components, n_neighbors = 30L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_neighbors	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

---

mb_pls *Multi-block PLS — formula entry point.*

Description

Multi-block PLS — formula entry point.

Usage

```
mb_pls(formula, data, ncomp = 2L, block_sizes, na.action = stats::na.omit)
```

Arguments

formula	A two-sided formula (e.g. 'y ~ .' or 'y ~ x1 + x2 + x3'). Response on the left, predictors on the right.
data	A 'data.frame' (or anything 'as.data.frame'-coercible) containing the response and predictor columns referenced by 'formula'.
ncomp	Integer. Number of latent components.
block_sizes	Integer vector summing to the number of predictors.
na.action	What to do with 'NA's. Default: 'na.omit'.

---

mb_pls_fit *Multi-block PLS (block-weighted SIMPLS).*

Description

Multi-block PLS (block-weighted SIMPLS).

Usage

```
mb_pls_fit(X, Y, n_components, block_sizes)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
block_sizes	Integer vector; must sum to ncol(X).

Value

A list with ‘coefficients‘, ‘predictions‘, ‘x_mean‘, ‘x_scale‘, ‘intercept‘, ‘block_weights‘, ‘rmse‘.

---

mir_pls *MIR-PLS — formula entry point.*

Description

MIR-PLS — formula entry point.

Usage

```
mir_pls(formula, data, ncomp = 2L, na.action = stats::na.omit)
```

Arguments

formula	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
data	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
ncomp	Integer. Number of latent components.
na.action	What to do with ‘NA‘s. Default: ‘na.omit‘.

Value

A ‘n4m_method_fit‘ object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use ‘predict()‘ for inference and ‘coef()‘ to extract regression coefficients.

`mir_pls_fit` *MIR-PLS — Multivariate Inverse Regression PLS.*

Description

MIR-PLS — Multivariate Inverse Regression PLS.

Usage

```
mir_pls_fit(X, Y, n_components)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_components`	Integer. Number of latent components.

Value

A list with ‘coefficients‘, ‘predictions‘, ‘x_mean‘, ‘y_mean‘, ‘rmse‘.

`missing_aware_nipals` *Missing-aware NIPALS — formula entry point.*

Description

Missing-aware NIPALS — formula entry point.

Usage

```
missing_aware_nipals(formula, data, ncomp = 2L, na.action = stats::na.pass)
```

Arguments

`formula`	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
`data`	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
`ncomp`	Integer. Number of latent components.
`na.action`	What to do with ‘NA‘s. Default: ‘na.omit‘.

Value

A ‘n4m_method_fit‘ object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use ‘predict()‘ for inference and ‘coef()‘ to extract regression coefficients.

`missing_aware_nipals_fit` *Missing-aware NIPALS PLS (Nelson 1996).*

Description

Missing-aware NIPALS PLS (Nelson 1996).

Usage

```
missing_aware_nipals_fit(X, Y, n_components)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_components`	Integer. Number of latent components.

`n_pls_fit` *N-PLS (3-way tensor) regression. 'X_flat' is the flattened (n, mode_j*mode_k) matrix.*

Description

N-PLS (3-way tensor) regression. 'X_flat' is the flattened (n, mode_j*mode_k) matrix.

Usage

```
n_pls_fit(X_flat, Y, n_components, mode_j, mode_k)
```

Arguments

`X_flat`	Numeric matrix. The flattened 3-way design tensor (rows = samples, cols = mode_j * mode_k).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_components`	Integer. Number of latent components.
`mode_j`	Integer. Size of the first non-sample tensor mode.
`mode_k`	Integer. Size of the second non-sample tensor mode.

n4m_abi_version *Loaded ABI version as an integer vector (major, minor, patch).*

Description

Loaded ABI version as an integer vector (major, minor, patch).

Usage

```
n4m_abi_version()
```

Value

An integer vector of length 3.

n4m_fit *Fit a PLS regression model via the libn4m C ABI.*

Description

Accepts a numeric matrix X (n x p) and a numeric vector or matrix Y (n x q). Both are coerced to double precision and row-major contiguous before being passed across the C boundary.

Usage

```
n4m_fit(
  X,
  Y,
  algo,
  n_components,
  store_scores = FALSE,
  center_x = TRUE,
  scale_x = TRUE,
  center_y = TRUE,
  scale_y = TRUE
)
```

Arguments

X	Numeric matrix, n x p.
Y	Numeric matrix or vector.
algo	Character. Solver name (see Details).
n_components	Integer >= 1.
store_scores	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

center_x	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
scale_x	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
center_y	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
scale_y	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

Details

'algo' selects the solver. Recognized values: "pls_nipals", "pls_orthogonal_scores", "pls_simpls", "pls_kernel_algorithm", "pls_wide_kernel", "pls_svd", "pls_power", "pls_randomized_svd", "pcr_svd", "opls_nipals".

Value

An external pointer wrapping the fitted model handle. Pass it to [n4m_predict()] to obtain predictions. The model is freed automatically when the external pointer is garbage-collected.

---

n4m_method *Low-level n4m method dispatcher.*

Description

Low-level n4m method dispatcher.

Usage

```
n4m_method(
  algo,
  X,
  Y,
  n_components,
  params = list(),
  center_x = TRUE,
  scale_x = FALSE,
  center_y = TRUE,
  scale_y = FALSE
)
```

Arguments

algo	Character; algorithm name (see Details).
X	Numeric matrix or NULL (for one_se_rule_compute).

`Y`	Numeric matrix or vector. Pass an n x 1 placeholder for classifier-style fits where labels go in 'params$y_labels'.
`n_components`	Positive integer.
`params`	Named list of algorithm-specific parameters (sparsity_lambda, sample_weights, block_sizes, X_target, y_labels, alpha_thresholds, ...).
`center_x`, `scale_x`, `center_y`, `scale_y`	Boolean preprocessing flags (default centering, no scaling — matches the Python tier-1 defaults).

Details

Supported algorithm names:

MethodResult fits (33): "sparse_simpls" "cppls" "ecr" "di_pls" "weighted_pls" "robust_pls" "ridge_pls" "continuum_regression" "recursive_pls" "n_pls" "kernel_pls" "o2pls" "sparse_pls_da" "group_sparse_pls" "fused_sparse_pls" "so_pls" "on_pls" "rosa" "bagging_pls" "boosting_pls" "random_subspace_pls" "gpr_pls" "pls_glm" "pls_qda" "pls_cox" "pds" "ds" "mir_pls" "missing_aware_nipals" "mb_pls" "lw_pls" "pls_lda" "pls_logistic"

Selectors (24): "spa_select" "cars_select" "interval_select" "stability_select" "uve_select" "random_frog_select" "scars_select" "ga_select" "pso_select" "vissa_select" "shaving_select" "bve_select" "t2_select" "wvc_select" "wvc_threshold_select" "emcuve_select" "randomization_select" "bipls_select" "sipls_select" "rep_select" "ipw_select" "st_select" "iriv_select" "irf_select" "vip_spa_select"

Diagnostics (2 via dispatcher; the other 2 stay in r_methods.c): "approximate_press_compute" "one_se_rule_compute"

Value

A named list with the MethodResult arrays + scalars. Index fields ('selected_indices', 'top_k_intervals', ...) are 1-based.

`n4m_predict` *Predict with a fitted n4m model.*

Description

Predict with a fitted n4m model.

Usage

```
n4m_predict(model, X)
```

Arguments

`model`	External pointer returned by [n4m_fit()].
`X`	Numeric matrix, n_new x p.

Value

Numeric matrix, n_new x n_targets.

`n4m_version` *Runtime version string of the loaded libn4m.*

Description

Runtime version string of the loaded libn4m.

Usage

```
n4m_version()
```

Value

A character scalar like "0.67.0+abi.1.1.0".

`o2pls` *O2-PLS — formula entry point (uses n_predictive for component count).*

Description

O2-PLS — formula entry point (uses n_predictive for component count).

Usage

```
o2pls(
  formula,
  data,
  n_predictive = 2L,
  n_x_orthogonal = 1L,
  n_y_orthogonal = 1L,
  na.action = stats::na.omit
)
```

Arguments

`formula`	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
`data`	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
`n_predictive`	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.

n_x_orthogonal	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
n_y_orthogonal	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
na.action	What to do with 'NA's. Default: 'na.omit'.

Value

A 'n4m_method_fit' object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use 'predict()' for inference and 'coef()' to extract regression coefficients.

`o2pls_fit` *O2-PLS (bi-directional OPLS).*

Description

O2-PLS (bi-directional OPLS).

Usage

```
o2pls_fit(X, Y, n_predictive = 2L, n_x_orthogonal = 1L, n_y_orthogonal = 1L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_predictive	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
n_x_orthogonal	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
n_y_orthogonal	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

`on_pls_fit` *OnPLS — Orthogonal multi-block PLS (joint + unique loadings).*

Description

OnPLS — Orthogonal multi-block PLS (joint + unique loadings).

Usage

```
on_pls_fit(X, Y, n_joint, n_unique_per_block, block_sizes)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_joint`	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
`n_unique_per_block`	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
`block_sizes`	Integer vector. Per-block feature counts for multi-block PLS.

`one_se_rule` *One-SE rule from a (max_components × n_folds) fold RMSE matrix.*

Description

One-SE rule from a (max_components × n_folds) fold RMSE matrix.

Usage

```
one_se_rule(fold_rmse_matrix)
```

Arguments

`fold_rmse_matrix`	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

opls — *Formula-based OPLS regression wrapper around the n4m C ABI.*

Description

Uses the same formula/S3 return contract as [pls()], with the model API configured as ‘algo = "opls_nipals"‘.

Usage

```
opls(formula, data, ncomp = 2L, na.action = stats::na.omit)
```

Arguments

formula	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
data	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
ncomp	Integer. Number of latent components.
na.action	What to do with ‘NA‘s. Default: ‘na.omit‘.

Value

An object of class ‘c("opls_fit", "n4m_fit", "pls_fit")‘.

pds_fit — *Piecewise Direct Standardization (calibration transfer).*

Description

Piecewise Direct Standardization (calibration transfer).

Usage

```
pds_fit(X_source, X_target, window_half_width = 2L)
```

Arguments

X_source	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
X_target	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
window_half_width	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.

pls *Formula-based PLS regression wrapper around the n4m C ABI*

Description

Tier-2 idiomatic R interface for tier-1 [n4m_fit()] / [n4m_predict()]. Provides a formula entry point and S3 generics so the returned object plays well with ‘predict()‘, ‘summary()‘, ‘coef()‘, ‘print()‘, base R ‘cv.glmnet‘-style workflows, and any caret model that supports a ‘predict(object, newdata)‘ adapter.

When the first argument is not a formula, ‘pls(x, y, ...)‘ dispatches to the matrix-oriented [pls_mdatools()] compatibility facade.

Usage

```
pls(
  formula,
  data,
  ncomp = 2L,
  algo = "pls_nipals",
  na.action = stats::na.omit,
  ...
)
```

Arguments

formula	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
data	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
ncomp	Integer. Number of latent components.
algo	Character. One of ‘"pls_nipals"‘, ‘"pls_simpls"‘, ‘"pls_svd"‘, ‘"pls_power"‘, ‘"pls_kernel_algorithm"‘, ‘"pls_wide_kernel"‘, ‘"pls_orthogonal_scores"‘, ‘"pls_randomized_svd"‘, ‘"pcr_svd"‘, ‘"opls_nipals"‘. Defaults to ‘"pls_nipals"‘ to match the R ‘pls‘ package’s default.
na.action	What to do with ‘NA‘s. Default: ‘na.omit‘.
...	For matrix-style calls, forwarded to [pls_mdatools()].

Value

An object of class ‘c("n4m_fit", "pls_fit")‘ with components: * ‘model‘ — external pointer to the libn4m model handle * ‘formula‘ — the call’s formula * ‘terms‘ — the ‘terms()‘ object describing the model * ‘xlevels‘ — factor levels of the predictors, for newdata coercion * ‘call‘ — the original call (for ‘print‘ / ‘summary‘) * ‘ncomp‘ — components used * ‘algo‘ — solver used * ‘n_features_in‘ — predictor count * ‘response_name‘ — left-hand side of the formula (string)

Examples

```
## Not run:
  set.seed(0)
  df <- data.frame(
    x1 = rnorm(100), x2 = rnorm(100), x3 = rnorm(100)
  )
  df$y <- 2 * df$x1 - df$x2 + rnorm(100, sd = 0.1)
  fit <- pls(y ~ ., data = df, ncomp = 3)
  summary(fit)
  preds <- predict(fit, newdata = df)

## End(Not run)
```

`pls_cox_fit` *PLS-Cox proportional hazards.*

Description

PLS-Cox proportional hazards.

Usage

```
pls_cox_fit(X, n_components, survival_times, event_indicators)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`n_components`	Integer. Number of latent components.
`survival_times`	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
`event_indicators`	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

`pls_diagnostics` *PLS diagnostics: T², Q, DModX from a fitted model.*

Description

PLS diagnostics: T², Q, DModX from a fitted model.

Usage

```
pls_diagnostics(model, X)
```

Arguments

model	External pointer from ‘n4m_fit()‘.
X	Numeric matrix to score (typically the training matrix).

Value

A list with ‘t2‘, ‘q‘, ‘dmodx‘ — each is a 1-row matrix of length nrow(X).

---

pls_glm *PLS-GLM — formula entry point. Default is Gaussian; set ‘family = "poisson"‘ for Poisson IRLS.*

Description

PLS-GLM — formula entry point. Default is Gaussian; set ‘family = "poisson"‘ for Poisson IRLS.

Usage

```
pls_glm(
  formula,
  data,
  ncomp = 2L,
  family = "gaussian",
  na.action = stats::na.omit
)
```

Arguments

formula	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
data	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
ncomp	Integer. Number of latent components.
family	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
na.action	What to do with ‘NA‘s. Default: ‘na.omit‘.

Value

A ‘n4m_method_fit‘ object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use ‘predict()‘ for inference and ‘coef()‘ to extract regression coefficients.

`pls_glm_fit` *PLS-GLM — Gaussian (default) or Poisson IRLS.*

Description

PLS-GLM — Gaussian (default) or Poisson IRLS.

Usage

```
pls_glm_fit(X, Y, n_components, poisson = FALSE)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_components`	Integer. Number of latent components.
`poisson`	Logical; TRUE selects the Poisson-link path.

Value

A list with ‘coefficients‘, ‘intercept‘, ‘predictions‘, ‘x_mean‘, ‘rmse‘.

`pls_lda_fit` *PLS-LDA — Linear Discriminant Analysis on PLS scores.*

Description

PLS-LDA — Linear Discriminant Analysis on PLS scores.

Usage

```
pls_lda_fit(X, y_labels, n_components, n_classes = NULL)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`y_labels`	Integer vector. Class labels.
`n_components`	Integer. Number of latent components.
`n_classes`	Integer >= 2. Number of class labels.

pls_logistic_fit *Multinomial logistic regression on PLS scores.*

Description

Multinomial logistic regression on PLS scores.

Usage

```
pls_logistic_fit(X, y_labels, n_components, n_classes = NULL)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
y_labels	Integer vector. Class labels.
n_components	Integer. Number of latent components.
n_classes	Integer >= 2. Number of class labels.

pls_mdatools *mdatools-style matrix PLS interface*

Description

Matrix-oriented PLS facade modelled after `mdatools::pls(x, y, ...)` for NIRS/chemometrics workflows. The top-level `pls(x, y, ...)` function dispatches here when its first argument is not a formula.

Usage

```
pls_mdatools(
  x,
  y,
  ncomp = min(nrow(x) - 1L, ncol(x), 20L),
  center = TRUE,
  scale = FALSE,
  cv = NULL,
  exclcols = NULL,
  exclrows = NULL,
  x.test = NULL,
  y.test = NULL,
  method = c("simpls", "cppls", "kernelpls",
             "widekernelpls", "oscorespls", "nipals", "pcr"),
  info = "",
  ncomp.selcrit = "min",
```

```
  lim.type = "ddmoments",
  alpha = 0.05,
  gamma = 0.5,
  cv.scope = "local",
  prep = NULL,
  fit_components = TRUE,
  ...
)
```

Arguments

x	Numeric predictor matrix.
y	Numeric response vector or matrix.
ncomp	Maximum number of components.
center	Logical; center predictors and response.
scale	Logical; standardize predictors and response.
cv	NULL, an integer number of folds, or a list of assessment row indices.
exclcols	Optional predictor columns to exclude.
exclrows	Optional rows to exclude before fitting.
x.test	Optional external test predictor matrix.
y.test	Optional external test response vector or matrix.
method	PLS/PCR algorithm selector.
info	Free-form model label.
ncomp.selcrit	Stored for compatibility.
lim.type	Stored for compatibility.
alpha	Stored for compatibility.
gamma	CPPLS gamma when method = "cppls".
cv.scope	Stored for compatibility.
prep	Optional preprocessing function applied to x and x.test.
fit_components	Logical; fit component prefixes 1:ncomp.
...	Reserved for compatibility.

Value

A n4m_mdatools_pls object with calibration results in calres; optional cross-validation and test results are stored in cvres and testres.

Examples

```
## Not run:
set.seed(1)
X <- matrix(rnorm(80), 20, 4)
y <- X[, 1] - X[, 2] + rnorm(20, sd = 0.05)
fit <- pls_mdatools(X, y, ncomp = 3, cv = 5)
```

```
predict(fit, x = X)
fit$calres$rmse
fit$cvres$rmse

## End(Not run)
```

`pls_monitoring` *PLS process monitoring (Hotelling T^2 + Q with alarms).*

Description

Computes phase-1 thresholds on ‘X_reference‘, then evaluates ‘X_monitor‘ against those thresholds and reports per-row alarms.

Usage

```
pls_monitoring(model, X_reference, X_monitor, alpha = 0.95)
```

Arguments

`model`	External pointer from ‘n4m_fit()‘.
`X_reference`	Phase-1 numeric matrix (used to set thresholds).
`X_monitor`	Phase-2 numeric matrix (rows are evaluated).
`alpha`	Confidence level (e.g. 0.95). Thresholds correspond to the (1 - alpha) quantile.

Value

A list with ‘t2‘, ‘q‘, ‘t2_alarms‘, ‘q_alarms‘, ‘any_alarms‘, ‘t2_threshold‘, ‘q_threshold‘.

`pls_qda_fit` *PLS-QDA (Quadratic Discriminant Analysis on PLS scores).*

Description

PLS-QDA (Quadratic Discriminant Analysis on PLS scores).

Usage

```
pls_qda_fit(X, y_labels, n_components)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`y_labels`	Integer vector. Class labels.
`n_components`	Integer. Number of latent components.

plsr *PLS package compatibility facade*

Description

These functions provide the familiar formula-oriented surface of the R **pls** package without importing it. Computation is backed by **n4m**; the facade is intended for NIRS/chemometrics scripts that already use `plsr()`, `pcr()`, `predict()`, `RMSEP()` and `selectNcomp()` patterns.

Usage

```
plsr(
  formula,
  ncomp = 2L,
  data,
  subset,
  na.action = stats::na.omit,
  method = c("kernelpls", "widekernelpls", "simpls",
             "oscorespls", "cppls", "nipals"),
  scale = FALSE,
  validation = c("none", "CV"),
  segments = 10L,
  center = TRUE,
  fit_components = TRUE,
  gamma = 0.5,
  sparsity_lambda = 0,
  ...
)

pcr(
  formula,
  ncomp = 2L,
  data,
  subset,
  na.action = stats::na.omit,
  method = c("svdpc", "pcr"),
  scale = FALSE,
  validation = c("none", "CV"),
  segments = 10L,
  center = TRUE,
  fit_components = TRUE,
  ...
)

mvr(
  formula,
  ncomp = 2L,
```

```
  data,
  subset,
  na.action = stats::na.omit,
  method = c("kernelpls", "widekernelpls", "simpls",
             "oscorespls", "cppls", "nipals", "pcr"),
  scale = FALSE,
  validation = c("none", "CV"),
  segments = 10L,
  center = TRUE,
  model = c("plsr", "pcr"),
  fit_components = TRUE,
  gamma = 0.5,
  sparsity_lambda = 0,
  ...
)

MSEP(object, ...)

RMSEP(object, ...)

R2(object, ...)

selectNcomp(object, method = c("min"), estimate = c("CV", "train"), ...)
```

Arguments

formula	A two-sided model formula.
ncomp	Number of latent components.
data	Data frame containing formula variables.
subset	Optional row subset.
na.action	Missing-value action.
method	PLS/PCR algorithm selector.
scale	Logical; standardize predictors and response.
validation	"none" or "CV".
segments	Number of contiguous CV segments, or a list of assessment row indices.
center	Logical; center predictors and response.
fit_components	Logical; fit component prefixes 1:ncomp for component-wise prediction and metrics.
gamma	CPPLS gamma.
sparsity_lambda	Sparse-SIMPLS soft threshold for the dense SIMPLS-compatible path.
model	"plsr" or "pcr".
object	A n4m_mvr object.
estimate	Metric source, "train" or "CV".
...	Reserved for compatibility.

Value

`plsr()`, `pcr()` and `mvr()` return a `n4m_mvr` object. `MSEP()`, `RMSEP()` and `R2()` return a small `mvrVal`-compatible list with a `val` array. `selectNcomp()` returns an integer component count.

Examples

```
## Not run:
set.seed(1)
df <- data.frame(x1 = rnorm(40), x2 = rnorm(40))
df$y <- 2 * df$x1 - df$x2 + rnorm(40, sd = 0.05)
fit <- plsr(y ~ ., data = df, ncomp = 2, method = "simpls")
predict(fit, df, ncomp = 1:2)
RMSEP(fit)
selectNcomp(fit)

## End(Not run)
```

`predict.n4m_fit` *Predict from a [pls()]-fitted model.*

Description

Predict from a [pls()]-fitted model.

Usage

```
## S3 method for class 'n4m_fit'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	A ‘n4m_fit‘ returned by [pls()].
`newdata`	A ‘data.frame‘ (or matrix). If ‘NULL‘, predictions on the training matrix are returned.
`...`	Ignored (for S3 generic compatibility).

Value

Numeric vector (for single-target regression) or matrix (multi-target).

predict.n4m_method_fit

Predict from a MethodResult-based n4m fit.

Description

Predict from a MethodResult-based n4m fit.

Usage

```
## S3 method for class 'n4m_method_fit'
predict(object, newdata = NULL, ...)
```

Arguments

object	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
newdata	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
...	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

pso_select

PSO-PLS (Binary Particle Swarm Optimization).

Description

PSO-PLS (Binary Particle Swarm Optimization).

Usage

```
pso_select(
  X,
  Y,
  n_components,
  n_swarm = 30L,
  n_iterations = 50L,
  w = 0.729,
  c1 = 1.494,
  c2 = 1.494,
  v_max = 4,
  seed = 0L
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_swarm	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
n_iterations	Integer >= 1. Number of outer-loop iterations.
w	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
c1	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
c2	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
v_max	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
seed	Integer. Random seed for reproducibility.

---

random_frog_select *Random Frog (Phase 5g).*

Description

Random Frog (Phase 5g).

Usage

```
random_frog_select(
  X,
  Y,
  n_components,
  n_iterations = 100L,
  initial_size = 30L,
  min_size = NULL,
  max_size = NULL,
  top_k = 10L,
  seed = 0L
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.

n_iterations	Integer >= 1. Number of outer-loop iterations.
initial_size	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
min_size	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
max_size	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
top_k	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
seed	Integer. Random seed for reproducibility.

`random_subspace_pls` *Random-subspace PLS — formula entry point.*

Description

Random-subspace PLS — formula entry point.

Usage

```
random_subspace_pls(
  formula,
  data,
  ncomp = 2L,
  n_estimators = 50L,
  features_per_subspace = 10L,
  seed = 0L,
  na.action = stats::na.omit
)
```

Arguments

formula	A two-sided formula (e.g. 'y ~ .' or 'y ~ x1 + x2 + x3'). Response on the left, predictors on the right.
data	A 'data.frame' (or anything 'as.data.frame'-coercible) containing the response and predictor columns referenced by 'formula'.
ncomp	Integer. Number of latent components.
n_estimators	Integer >= 1. Number of bootstrap / boosting / random-subspace estimators.
features_per_subspace	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
seed	Integer. Random seed for reproducibility.
na.action	What to do with 'NA's. Default: 'na.omit'.

Value

A ‘n4m_method_fit‘ object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use ‘predict()‘ for inference and ‘coef()‘ to extract regression coefficients.

`random_subspace_pls_fit` *Random-subspace PLS (Ho 1998).*

Description

Random-subspace PLS (Ho 1998).

Usage

```
random_subspace_pls_fit(
  X,
  Y,
  n_components,
  n_estimators = 50L,
  features_per_subspace = 10L,
  seed = 0L
)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_components`	Integer. Number of latent components.
`n_estimators`	Integer >= 1. Number of bootstrap / boosting / random-subspace estimators.
`features_per_subspace`	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
`seed`	Integer. Random seed for reproducibility.

randomization_select *Randomization test selector.*

Description

Randomization test selector.

Usage

```
randomization_select(
  X,
  Y,
  n_components,
  n_permutations = 100L,
  randomization_seed = 0L,
  alpha = 0.05
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_permutations	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
randomization_seed	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
alpha	Numeric in [0, 1]. Elastic-net / penalty mixing parameter.

recursive_pls *Recursive PLS – formula entry point.*

Description

Recursive PLS stores in-sample predictions rather than a reusable coefficient model. 'predict()' therefore accepts only the fitted training design.

Usage

```
recursive_pls(
  formula,
  data,
  ncomp = 2L,
  window_size = 50L,
  na.action = stats::na.omit
)
```

Arguments

formula	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
data	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
ncomp	Integer. Number of latent components.
window_size	Integer moving-window size.
na.action	What to do with ‘NA‘s. Default: ‘na.omit‘.

---

recursive_pls_fit *Moving-window recursive PLS.*

Description

Moving-window recursive PLS.

Usage

```
recursive_pls_fit(X, Y, n_components, window_size = 50L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
window_size	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.

---

rep_select *REP-PLS.*

Description

REP-PLS.

Usage

```
rep_select(
  X,
  Y,
  n_components,
  n_steps = 10L,
  min_features = 5L,
  remove_count = 1L
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_steps	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
min_features	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
remove_count	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

---

ridge_pls *Ridge PLS — formula entry point.*

Description

Ridge PLS — formula entry point.

Usage

```
ridge_pls(
  formula,
  data,
  ncomp = 2L,
  ridge_lambda = 1,
  na.action = stats::na.omit
)
```

Arguments

formula	A two-sided formula (e.g. 'y ~ .' or 'y ~ x1 + x2 + x3'). Response on the left, predictors on the right.
data	A 'data.frame' (or anything 'as.data.frame'-coercible) containing the response and predictor columns referenced by 'formula'.
ncomp	Integer. Number of latent components.
ridge_lambda	Numeric >= 0. Ridge regularisation strength.
na.action	What to do with 'NA's. Default: 'na.omit'.

Value

A 'n4m_method_fit' object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use 'predict()' for inference and 'coef()' to extract regression coefficients.

`ridge_pls_fit` *L2-augmented PLS regression.*

Description

L2-augmented PLS regression.

Usage

```
ridge_pls_fit(X, Y, n_components, ridge_lambda = 1)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_components`	Integer. Number of latent components.
`ridge_lambda`	Numeric >= 0. Ridge regularisation strength.

`robust_pls` *Robust PLS — formula entry point.*

Description

Robust PLS — formula entry point.

Usage

```
robust_pls(
  formula,
  data,
  ncomp = 2L,
  huber_k = 1.345,
  max_irls_iter = 20L,
  na.action = stats::na.omit
)
```

Arguments

`formula`	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
`data`	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
`ncomp`	Integer. Number of latent components.
`huber_k`	Numeric >= 0. Huber loss tuning constant.

max_irls_iter	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
na.action	What to do with 'NA's. Default: 'na.omit'.

Value

A 'n4m_method_fit' object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use 'predict()' for inference and 'coef()' to extract regression coefficients.

`robust_pls_fit` *Robust PLS via Huber IRLS.*

Description

Robust PLS via Huber IRLS.

Usage

```
robust_pls_fit(X, Y, n_components, huber_k = 1.345, max_irls_iter = 20L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
huber_k	Numeric >= 0. Huber loss tuning constant.
max_irls_iter	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

`rosa_fit` *ROSA — Response-Oriented Sequential Alternation.*

Description

ROSA — Response-Oriented Sequential Alternation.

Usage

```
rosa_fit(X, Y, n_components, block_sizes)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
block_sizes	Integer vector. Per-block feature counts for multi-block PLS.

scars_select *SCARS — Stability + CARS.*

Description

SCARS — Stability + CARS.

Usage

```
scars_select(
  X,
  Y,
  n_components,
  n_iterations = 50L,
  min_features = 5L,
  sample_fraction = 0.8,
  seed = 0L
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_iterations	Integer >= 1. Number of outer-loop iterations.
min_features	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
sample_fraction	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
seed	Integer. Random seed for reproducibility.

selectivity_ratio_select *Selectivity-ratio ranker.*

Description

Selectivity-ratio ranker.

Usage

```
selectivity_ratio_select(model, X, top_k)
```

Arguments

model	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
X	Numeric matrix used for the fit (re-passed for diagnostics).
top_k	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

Value

A list with 'scores' and 'selected_indices'.

---

shaving_select *Shaving selector.*

Description

Shaving selector.

Usage

```
shaving_select(
  X,
  Y,
  n_components,
  n_steps = 10L,
  min_features = 5L,
  shave_fraction = 0.1
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_steps	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
min_features	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
shave_fraction	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

sipls_select *siPLS — synergistic interval PLS.*

Description

siPLS — synergistic interval PLS.

Usage

```
sipls_select(X, Y, n_components, interval_width = 10L, combination_size = 2L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
interval_width	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
combination_size	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

snv_transform *Portable preprocessing transforms*

Description

Thin R wrappers over libn4m's preprocessing C ABI. The R layer performs matrix layout conversion only; all numerical work is delegated to libn4m.

Usage

```
snv_transform(X, with_mean = TRUE, with_std = TRUE, ddof = 0L)

savgol_transform(
  X,
  window_length,
  polyorder = 3L,
  deriv = 0L,
  delta = 1,
  mode = "mirror",
  cval = 0
)
```

Arguments

X	Numeric matrix.
with_mean	Logical; center each row before scaling.
with_std	Logical; scale each row by its standard deviation.
ddof	Integer degrees-of-freedom correction.
window_length	Odd integer Savitzky-Golay window length.
polyorder	Savitzky-Golay polynomial order.
deriv	Savitzky-Golay derivative order.
delta	Sample spacing.
mode	Boundary mode: "mirror", "constant", "nearest", "wrap", or "interp".
cval	Constant fill value used when mode = "constant".

Value

Numeric matrix with the same dimensions as X.

---

so_pls_fit *Sequential & Orthogonalised multi-block PLS (Næs et al. 2011). 'block_sizes' integer vector summing to ncol(X); 'n_components_per_block' integer vector of same length.*

Description

Sequential & Orthogonalised multi-block PLS (Næs et al. 2011). 'block_sizes' integer vector summing to ncol(X); 'n_components_per_block' integer vector of same length.

Usage

```
so_pls_fit(X, Y, block_sizes, n_components_per_block)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
block_sizes	Integer vector. Per-block feature counts for multi-block PLS.
n_components_per_block	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

spa_select *SPA — Successive Projections Algorithm.*

Description

Fit-data selector: pass (X, Y) and the desired number of features.

Usage

```
spa_select(X, Y, n_components, top_k)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
top_k	Number of features to select.

Value

A list with ‘selected_indices‘ (1-based) and ‘best_rmse‘.

sparse_pls *Sparse SIMPLS — formula entry point.*

Description

Sparse SIMPLS — formula entry point.

Usage

```
sparse_pls(
  formula,
  data,
  ncomp = 2L,
  sparsity_lambda = 0.05,
  na.action = stats::na.omit
)
```

Arguments

formula	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
data	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
ncomp	Integer. Number of latent components.
sparsity_lambda	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
na.action	What to do with ‘NA‘s. Default: ‘na.omit‘.

Value

A ‘n4m_method_fit‘ object carrying the fitted model, in-sample predictions, training RMSE, and method-specific metadata. Use ‘predict()‘ for inference and ‘coef()‘ to extract regression coefficients.

---

sparse_pls_da_fit *Sparse PLS-DA classifier (‘y_labels‘ is an integer vector of class IDs).*

Description

Sparse PLS-DA classifier (‘y_labels‘ is an integer vector of class IDs).

Usage

```
sparse_pls_da_fit(X, y_labels, n_components, sparsity_lambda = 0.05)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
y_labels	Integer vector. Class labels.
n_components	Integer. Number of latent components.
sparsity_lambda	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.

`sparse_simpls_fit` *Sparse SIMPLS fit.*

Description

Sparse SIMPLS fit.

Usage

```
sparse_simpls_fit(X, Y, n_components, sparsity_lambda = 0.05)
```

Arguments

`X`	Numeric matrix.
`Y`	Numeric matrix or vector.
`n_components`	Integer >= 1.
`sparsity_lambda`	Soft-threshold magnitude per component (>= 0).

Value

A list with ‘coefficients‘, ‘predictions‘, ‘x_mean‘, ‘y_mean‘, ‘weights_w‘, ‘rmse‘.

`st_select` *ST-PLS — score-threshold selector.*

Description

ST-PLS — score-threshold selector.

Usage

```
st_select(X, Y, n_components, thresholds, min_selected = NULL)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_components`	Integer. Number of latent components.
`thresholds`	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
`min_selected`	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.

stability_select *Stability selector (coefficient stability, MCUVE-style).*

Description

Stability selector (coefficient stability, MCUVE-style).

Usage

```
stability_select(X, Y, n_components, top_k = 10L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
top_k	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

t2_select *T2-PLS - sweep over alpha thresholds.*

Description

T2-PLS - sweep over alpha thresholds.

Usage

```
t2_select(X, Y, n_components, alpha_thresholds, min_selected = NULL)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
alpha_thresholds	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
min_selected	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

uve_select *UVE — Uninformative Variable Elimination.*

Description

UVE — Uninformative Variable Elimination.

Usage

```
uve_select(X, Y, n_components, noise_features = NULL, noise_seed = 0L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
noise_features	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
noise_seed	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

vip_select *VIP (Variable Importance in Projection) ranker.*

Description

Operates on an already-fitted n4m model handle (from 'n4m_fit'). Returns the top-'top_k' features by VIP score.

Usage

```
vip_select(model, X, top_k)
```

Arguments

model	External pointer from 'n4m_fit()'.
X	Numeric matrix used for the fit (re-passed for diagnostics).
top_k	Integer; number of features to return.

Value

A list with 'scores' (length p VIP scores) and 'selected_indices' (1-based, length top_k).

vip_spa_select *VIP-SPA hybrid selector.*

Description

VIP-SPA hybrid selector.

Usage

```
vip_spa_select(X, Y, n_components, vip_threshold = 0.3, top_k = 10L)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
vip_threshold	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
top_k	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

vissa_select *VISSA — Variable Iterative Space Shrinkage Approach.*

Description

VISSA — Variable Iterative Space Shrinkage Approach.

Usage

```
vissa_select(
  X,
  Y,
  n_components,
  n_iterations = 20L,
  n_submodels = 100L,
  ratio_kept = 0.1,
  threshold = 0.5,
  floor_probability = 0.01,
  seed = 0L
)
```

Arguments

X	Numeric matrix of predictors (rows = samples, cols = features).
Y	Numeric matrix or vector of responses, with one row per sample.
n_components	Integer. Number of latent components.
n_iterations	Integer >= 1. Number of outer-loop iterations.
n_submodels	Integer >= 1. Number of inner sub-models per VISSA-style iteration.
ratio_kept	Numeric in (0, 1]. Fraction of features kept per iteration.
threshold	Numeric. Convergence / pruning threshold.
floor_probability	Numeric in [0, 0.5). Lower bound on per-feature retention probability.
seed	Integer. Random seed for reproducibility.

---

weighted_pls *Sample-weighted PLS — formula entry point.*

Description

Sample-weighted PLS — formula entry point.

Usage

```
weighted_pls(formula, data, ncomp = 2L, weights, na.action = stats::na.omit)
```

Arguments

formula	A two-sided formula (e.g. ‘y ~ .‘ or ‘y ~ x1 + x2 + x3‘). Response on the left, predictors on the right.
data	A ‘data.frame‘ (or anything ‘as.data.frame‘-coercible) containing the response and predictor columns referenced by ‘formula‘.
ncomp	Integer. Number of latent components.
weights	Numeric vector of length nrow(data) with sample weights.
na.action	What to do with ‘NA‘s. Default: ‘na.omit‘.

`weighted_pls_fit` *Sample-weighted PLS (sqrt(w)-prescaled SIMPLS).*

Description

Sample-weighted PLS (sqrt(w)-prescaled SIMPLS).

Usage

```
weighted_pls_fit(X, Y, n_components, sample_weights)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_components`	Integer. Number of latent components.
`sample_weights`	Numeric vector of length nrow(X), strictly positive finite weights.

Value

A list with ‘coefficients‘, ‘predictions‘, ‘x_mean‘, ‘y_mean‘, ‘rmse‘.

`wvc_select` *WVC-PLS — weighted vector correlation top-k selector.*

Description

WVC-PLS — weighted vector correlation top-k selector.

Usage

```
wvc_select(X, Y, n_components, top_k = 10L, normalize = TRUE)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_components`	Integer. Number of latent components.
`top_k`	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.
`normalize`	Method-specific parameter. See the underlying ‘*_fit()‘ function for the exact semantics.

`wvc_threshold_select` *WVC-threshold selector.*

Description

WVC-threshold selector.

Usage

```
wvc_threshold_select(
  X,
  Y,
  n_components,
  normalize = TRUE,
  threshold = 0,
  threshold_factor = 1,
  min_selected = 1L
)
```

Arguments

`X`	Numeric matrix of predictors (rows = samples, cols = features).
`Y`	Numeric matrix or vector of responses, with one row per sample.
`n_components`	Integer. Number of latent components.
`normalize`	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
`threshold`	Numeric. Convergence / pruning threshold.
`threshold_factor`	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.
`min_selected`	Method-specific parameter. See the underlying '*_fit()' function for the exact semantics.

Index